

Research on pureXML Performance Analysis

Zhao Lili¹, Gao Zhen^{1*}, Yang Jian², Wang Min¹, Li Xiaofeng³

¹School of Software Engineering, Tongji University, Shanghai, China,

²Software School, Fudan University, Shanghai, China

³IBM CDL, Beijing, China

^{1*}gaozhen@tongji.edu.cn, ²yangjian@fudan.edu.cn

Abstract—DB2 integrates the relational database and XML, implements the combination of relational engine and hierarchical engine. Pure XML has expansive application future. Most of the current mainstream relational databases integrate XML processing capacity. However, different DBMS adopts different framework and implementation method in processing XML data. This results in different XML processing capacity. DB2 adopted pureXML to store XML data in the newest version. This paper focuses on the research of pureXML technology and compares its performance with the current popular database product Oracle and SQL Server. Detailed lab materials and results are given to provide the useful references for choosing proper relational databases to process XML data in practice.

Keywords—pureXML; DB2; performance analysis; storage efficiency

I. INTRODUCTION

So far, the relational database has got to the heart of company storage and data application. Under the background of WEB2.0, SOA also added fuel to the flames. XML is becoming an important public language in data exchange for its simplicity in reading and writing data in any application program. The advent of DB2 V9 pureXML has overtaken the regional enteritis of relational database and opened a new area for database application and development^[1]. It not just implements XML Database Query, XML Data Storage and XML data management, but also improves data analysis precision^[7]. It met with a favorable reception in market and got lots of successful applications in Healthcare, government, educational Publishing House and information collections. This paper will do research on pureXML technology, do analysis and test for pureXML in performance area through compare with other databases.

At present, the explosive increase of XML data and the Web Service accessing this kind of data, make the enterprises create more and more new-style information architecture. In this process, XML data storage becomes a very critical component. These new XML storage technology needs in E-Tax system, electronic medical records system and other new fields have been obvious questions against current XML storage technology. However, the traditional XML data storage technology is difficult to meet these requirements. Under the circumstance, pureXML technology and a new hybrid database - DB2 V9 have emerged, they beyond the inherent limitations of relational database and its XML storage capability. The technology have opened up a new scientific field for XML database application and development, thanks to hierarchical data organization of pureXML, It's easier for us to molding data

model, both Relational and hierarchical data model. As a result, it is foreseen that it has a bright future applying pureXML. This article is aimed to analyze and research DB2 pureXML technology, discuss how DB2 pureXML store and manage the contents of XML in an efficient way. On the basis of analyzing pureXML's implementation mechanism, test pureXML performance, compare this test results with some other common XML databases, and then analyze pureXML performance more deeply.

II. PUREXML IMPLEMENT MECHANISM

A. Why PureXML

Before DB2 version 9, the traditional XML stores data with the following ways:

- 1) Store XML document in the file systems;
- 2) Store XML data in the LOB of relational database;
- 3) Decompose the XML data element and store them in the multiple tables and columns.
- 4) Regard the XML as a pure database system

However, those ways still can't meet the performance requirement. The file system is fit for simple task. It can't meet the requirement of large documents, and is hard to realize the concurrency, security and usability. The DBMS way can relieve some problems of file system, but still has some limitation in XML data management [2].

PureXML is a new feature of DB2, it can store XML data in the table with the original XML format. pureXML technique is not only an external interface, it also extends to the kernel of DB2 engine. DB2 V9 integrates XML and relational services, provides a totally new mixed server to industry.

DB2 V9 pureXML technique includes the following functions:

- 1) PureXML data type and storage technique can manage hierarchical structure of XML document efficiently.
- 2) PureXML indexical technique can locate the sub tree of XML document quickly.
- 3) Base on the new query language (XQuery and SQL/XML) and new query optimizing technique of industry.
- 4) Support to manage, verify and evolve XML mode.
- 5) Integrate the popular API and develop environment.

6) XML's decompose and release function which meets the current relational mode

7) Keep the reliability, usability, scalability, performance, security and Maturity of DB2.

Figure 1 has simply shown the architecture of hybrid database.

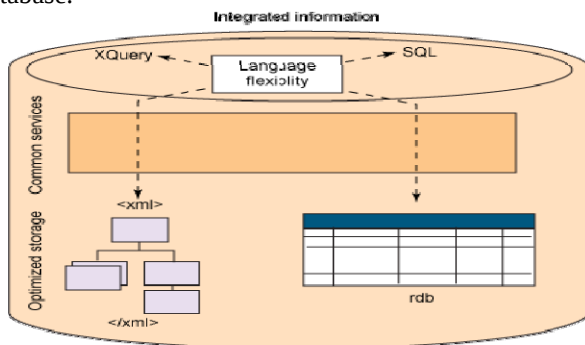


Figure 1. Architecture of Hybrid Database

B. PureXML storage mechanism

PureXML is a new technology to store the XML data in DB2 version 9, it totally discards the traditional full storage and decomposition storage technology which both have obvious limitations in function and performance. XML data is formatted into data buffer page so it can provide fast navigation, quick selection and simple index mechanism. The minimal storage unit is stored with Tree-Structure XML data which is similar with Tree-Structure data parsed with DOM. Actually, XML document is parsed with SAX when it is inserted into DB2 version 9, and the searching effect is as convenient as DOM tree. Figure 2 has simply shown the storage mechanism of pureXML.

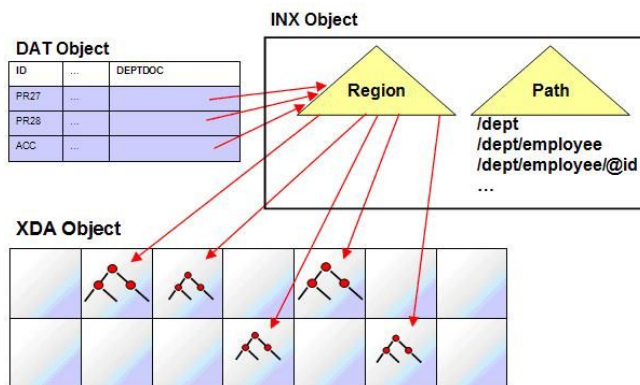


Figure 2. PureXML overall Storage Mechanism

C. PureXML query mechanism

DB2 pureXML provide efficient and versatile function to support XML data management. DB2 process XML according to the XML inherent attribute. DB2 provides two kinds of languages: pure SQL and SQL/XML. You can use SQL or SQL embed XQuery. It ensures the function to

access relational data, XML data or both. You can choose either way to query XML data.

1. Common SQL
2. SQL/XML which embeds XQuery in SQL

Figure 3 has simply shown overview of pureXML query mechanism.

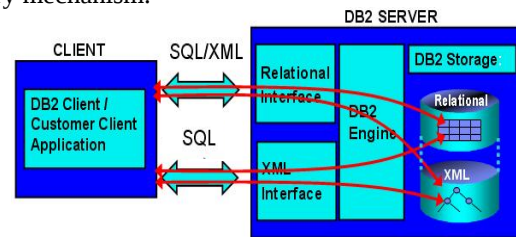


Figure 3. Overview of PureXML Query Mechanism

D. PureXML index mechanism

Traditional relational index can support combinatorial index which assembled by one or more indexes, but XML index only support the index which stores in single XML column.

It is not rare to see a XML program which manages millions of XML documents. So it needs large numbers of XML indexes to improve query performance. DB2 supports to build specific index on path, so the element and attribute are usually as predication.

New XML index can evaluate XML mode expression efficiently which improve query performance of XML document. Compared with traditional relational indexes, the traditional index are composed by one or more columns while XML value index uses specific XML mode expression to compile value index of path and XML document. If the value is not specified in the document, the index can be filled with default attribute and elements through inserting pattern.

Similar to relational index, XML index also created with SQL DDL statements and CREATE INDEX statements. However, besides specifying the object column for index, user should also specify XMLPattern to ensure the XML document segment you interested.

Figure 4 has simply shown the way to creating anPureXML index.

III. PUREXML PERFORMANCE TEST PREPARATION

For pureXML has implemented the native support for XML, DB2 V9 is no longer just a relational database, it becomes a mixed database which both support the relational data model and hierarchical data model. on the one hand, It inherits the old two-dimensional database technology to support the standard SQL inquiries, but on the other hand, it has a well-found XML native technology to support the standard XQuery inquiries. At the same time, two-dimensional database technology and hierarchical XQuery inquiries can be used in a nested mutually way [6].

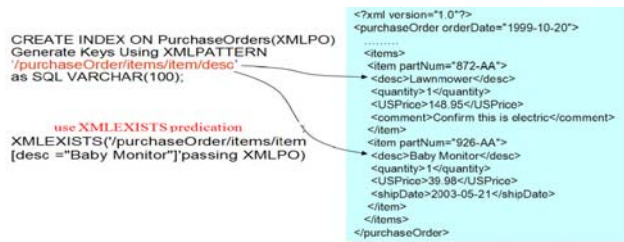


Figure 4. Creating a PureXML index

A. Performance Test Object, Tools and System Configuration

- Test objects: DB2 v9.5, Oracle 11g, SQL Server 2008. The three database Systems are Enterprise Edition.
- Test tool: TPoX (Transaction Processing over XML).

After comparison, we will choose TPoX which stands for Transaction Processing over XML. TPoX tests the global performance of XML data processing capacity, it is different with other testing tools which only test on execute efficiency for XQuery. Partial testing data of TPox come from actual environment. For the high modeling, it is used to evaluate XML database performances like XQuery, SQL/XML, XML storage, XML index, XML Schema verification, XML updating ,concurrency operations and so on [3].

- System configuration: In order to separate the client workload driver from the database server we used two Intel based Xeon machines connected by a dedicated gigabit network. The client workload driver was running on a Linux machine that never exceeded 25% total CPU utilization in any of the tests. The database server utilized an Intel server running Windows 2003 R2 Server. In various tests, simulations with 10 users gave the best overall average performance for each database vendor. System configuration for XML performance testing is shown as figure 5.

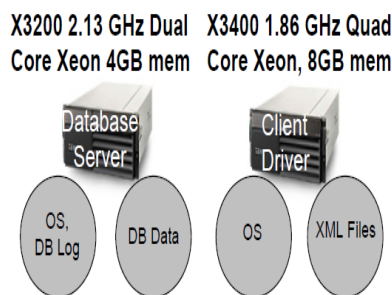


Figure 5. System Configuration for XML Performance Testing

B. Performance Test Model

TPoX data model include two business entities: customer and brokerage house. Just as figure6 shows, customers buy and sell the securities through orders, brokerage house process the transactions according to the requests from customers. The system core is a database that supports the

XML functions, and its performance determines the performance of this application.

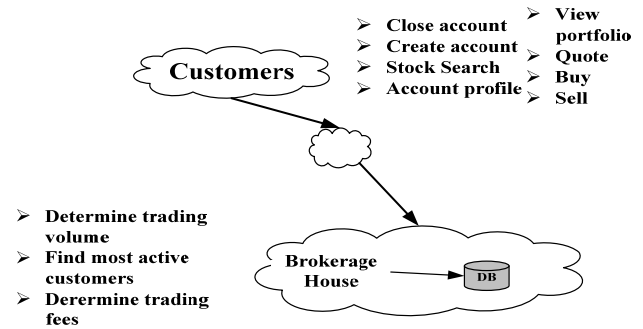


Figure 6. Basic Data Model of TPoX

Figure 7 is a sample of the FIXML financial data used during the transactions.

```
<FIXML>
  <NewOrdSingle C1OrdID="123456"
    Side="2"
    TransactIm="2003-06-15T01:14:49-05:00"
    OrderType="2"
    Price="93.25"
    Acct="26522154">
    <Header Sent="2001-06-21T01:31:28-05:00"
      PosDup="N"
      PosRsd="N"
      SeqNum="521">
      <Sender ID="AFUNDMGR"/>
      <Target ID="ABROKER"/>
    </Header>
    <Instrument Symbol="IBM"
      ID="459200101"
      IDSrc="1"/>
    <OrderQuantity Qty="1000" Cur="USD"/>
  </NewOrdSingle>
</FIXML>
```

Figure 7. FIXML Financial Data

C. Performance Test Data

During the load we inserted 20,833 securities, 60,000 customer accounts, and 300,000 order records for approximately 1GB XML data. The data model Entity-Relationship is shown in the following figure 8.

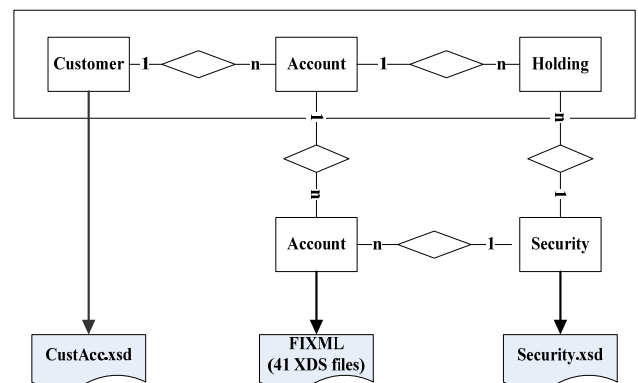


Figure 8. Entity-Relationship Chart of Test Data Model

Customer has a XML document which includes the personal information like name, address, primary customer status and account information (holding security and balance).

Order has a XML document for each transaction which includes account id, transaction date, securities symbol, transaction volume and so on.

Securities have a XML document for each kind of negotiable securities which are transacted in the market. The XML document includes securities symbol, securities type, securities price, securities sector.

The following table 1 is the attributes of the above entities.

TABLE I. TABLE STRUCTURE OF TEST DATA

entity	attribute
Customer	Customer_id, name, address, email, password, date_of_birth
Account	account_id, customer_id, account_nr, balance
Securities	symbol, company, volume, price, sector
Holding	symbol, quantity, price
Order	order_id, account_id, symbol, order_type, status, date, price, fee, quantity

IV. PUREXML PERFORMANCE TEST

In this section we will do performance test for DB2, Oracle and SQL Server database which all support XML. Firstly, testing on XML data stored without any indexes, then add index to test. The result shows that pureXML of DB2 has the best performance.

A. Simple XML Data

For XML data stored without any indexes, DB2 was the fastest at both loading and querying XML. SQL Server was able insert XML data almost as fast as DB2, but took significantly longer to query XML data. A big surprise was Oracle, which was significantly slower at inserting XML data and dramatically slower at querying XML. This showed that DB2's performance is much better than either Microsoft or Oracle, even straight out of the box without requiring a DBA to tune the system [8].

DB2 inserts XML data just slightly faster (1.02x) and queries XML data 2.5x faster than SQL Server. DB2 inserts XML data 2.5x faster and queries XML data 17x faster than Oracle.

Test result is shown in the following figure9 and figure10.

B. Using Indexes

Indexes are commonly used to improve database query performance. However adding indexes can also increase processing overhead resulting in longer insert and update times as well as increasing the amount of disk space used. Most data is queried many more times than it is inserted, making index performance critical to overall database performance.

DB2, Oracle, and SQL Server have very similar mechanisms for defining indexes on traditional relational databases. Relational database index definitions specify a column or group of columns to create the index on. For example if you accessed the customer by their customer id,

the DBA might want to index the id column of the customer table to improve query performance.

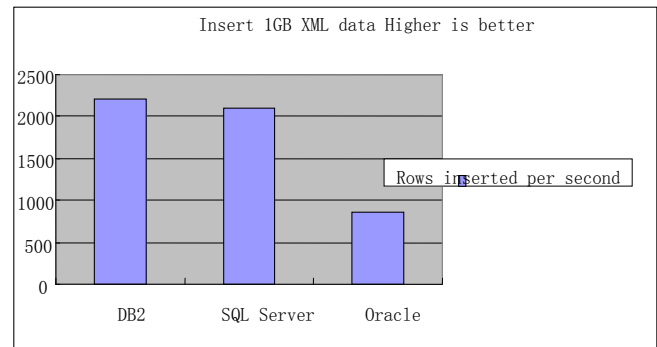


Figure 9. The Comparison of Insert Speed for Simple XML Data

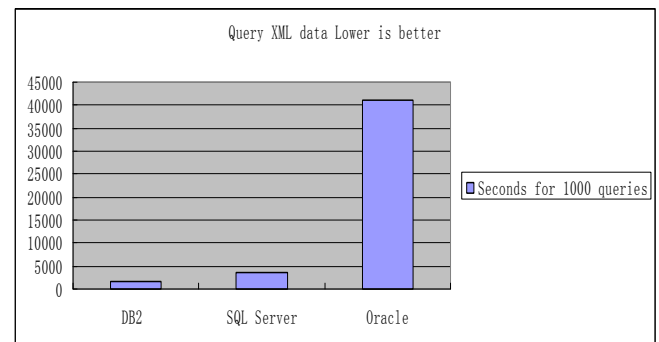


Figure 10. The Comparison of Query Speed for Simple XML Data

DB2, Microsoft, and Oracle chose different approaches to indexing XML data. DB2 and Oracle indexes are built using XPath statements (part of the XQuery standard). XPath is a language for selecting “nodes” within an XML document and is very flexible. For example to find all the customer ids the following XPATH statement could be used: `//customer/@id`. This expression returns all the customers id attributes within the XML document [4].

Microsoft took a much different approach with SQL Server, using “primary” and “secondary” indexes. The primary XML index “indexes all tags, values, and paths over the XML instances in an XML column”. This index alone can be quite large, it can actually be many times the size of the XML data it references. Once you have a primary index you can also define different types of secondary indexes based on what part of the XML you want to search for. The closest equivalent to SQL Servers primary index using XPath would be to define an index on everything within the XML document [5].

In the second performance scenario we added a full path index (`/`) on the XML data. This was the minimum index permissible with SQL Server, and was easily defined for DB2 and Oracle. For the third scenario indexes that targeted specific XML elements were used. For DB2 and Oracle this was done with partial indexes using XPATH's pointing to elements used in the XQueries. For SQL Server secondary indexes were defined.

C.XML data with a full path index

As explained earlier full path indexes may negatively impact insert performance and increase storage overhead, but they were necessary in order to accommodate the way SQL Server XML handles indexes. Even with the overhead of indexing every path, element, and attribute in the XML document, DB2 was the fastest at both loading and querying XML, incurring the least overall impact on insert times while obtaining the best improvements in query times.

SQL Server's insert performance was 4.8x slower than when inserting non-indexed XML data. And contrary to expectations, query response times actually more than doubled when using a primary index.

Using a full path index (//*) had a huge negative impact on Oracle insert performance, which took over 63x times as long as when inserting non-indexed XML. While Oracle query times improved, they were still almost 3.5x slower than DB2's with full path indexing, and even slower than DB2 without indexes.

DB2 inserts XML data 2.6x faster and queries XML data over 6x faster than SQL Server when full path indexes are defined. DB2 inserts XML data 87x faster and queries XML data 3.5x faster than Oracle when full path indexes are defined.

Test result is shown in the following figure 11 and figure 12.

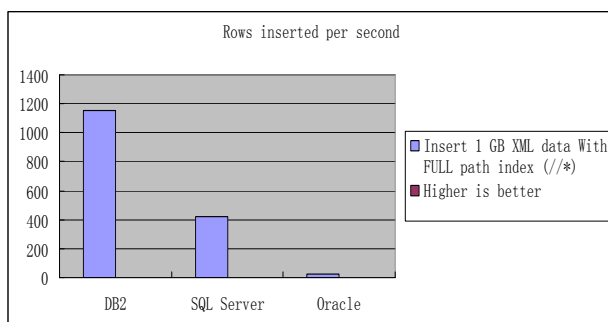


Figure 11. Insert speed comparison for XML data with full path index

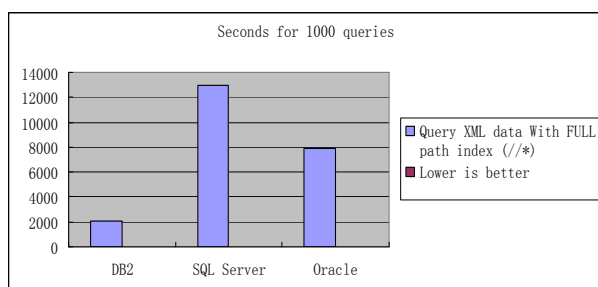


Figure 12. Query speed comparison for XML data with full path index

D.XML data with a partial path index

With DB2 and Oracle using XPath, database administrators can define indexes on XML data for only that portion of the XML data that is important to you (herein referred to as a "partial path"). To accomplish a similar result

with SQL Server both primary and secondary indexes must be defined [9].

Again, DB2 derived the most benefit from using indexes with the fastest rate of both XML insertion and XML queries.

Using both primary and secondary indexes with SQL Server XML data impacted load times even more. SQL Server took over 185x as long to insert XML data with primary and secondary indexes than without indexes. SQL Server query performance did improve with both primary and secondary indexes, but not as much as DB2, and was 6x slower than DB2 querying XML data.

Although the negative impact of partial indexes on Oracle was much less than full path indexes, inserting XML still took 2.7x as long with partial indexes than without indexes and the average Oracle query time still went up instead of down showing a negative impact instead of an improvement.

The addition of indexes on the XML data improved DB2's performance more than any of the competitors. This showed that DB2 benefits more from the use of XML indexes than either SQL Server or Oracle.

DB2 inserts XML data over 160x faster and queries XML data 6x faster than SQL Server when partial indexes are used. DB2 inserts XML data 6x faster and queries XML data 77x faster than Oracle when partial indexes are used.

Test result is shown in the following figure13 and figure14.

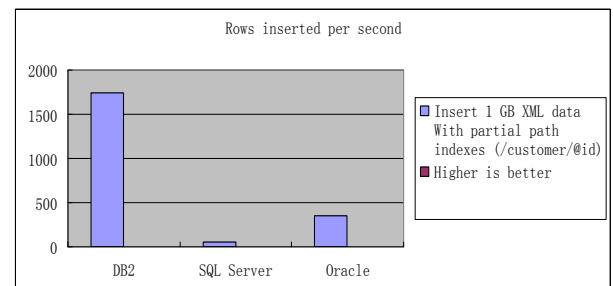


Figure 13. Insert speed comparison for XML data with partial path index

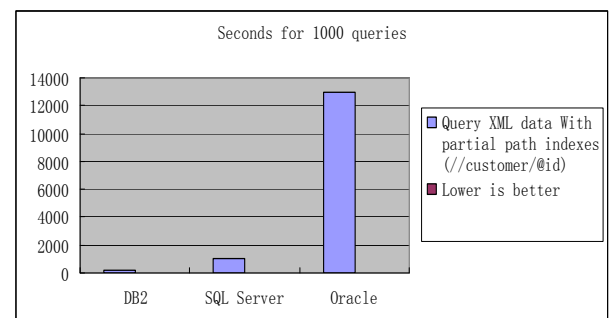


Figure 14. Query speed comparison for XML data with partial path index

E.XML data storage efficiency

XML was designed as a flexible, self-describing, human readable data format. While easily understandable, this lead

to an expensive storage model with lots of repeating information (such as XML element and attribute names). The growth of XML highlights the need to consider how databases store XML when evaluating their overall performance. DB2's pureXML was designed to both store and query XML efficiently [10].

DB2 uses significantly less storage than SQL Server, from 1/3 less for simple XML to an amazing 92% less when using partial indexes compared to secondary indexes.

DB2 requires about the same amount of storage as Oracle for XML with partial indexes. However it requires about 50% less than Oracle when full path indexes are used.

Test result is shown in the following figure15 and figure16.

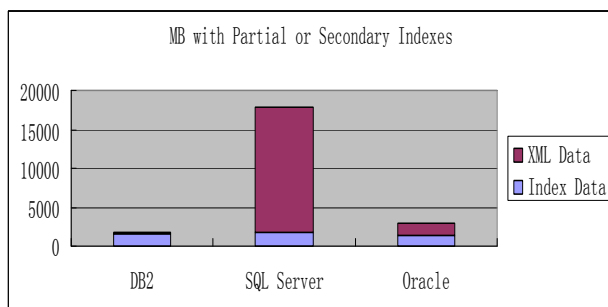


Figure 15. Storage usage comparison for XML data with partial path index

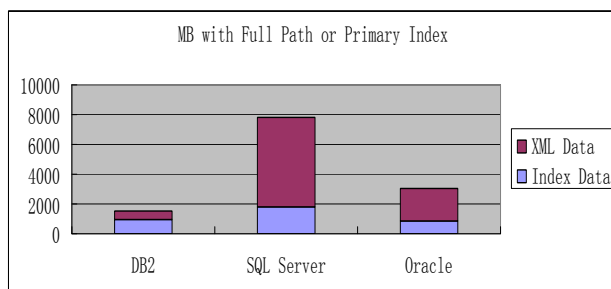


Figure 16. Storage usage comparison for XML data with full path index

V.CONCLUSION

The combination of relational databases with XML technology has rapidly become one of the key technologies for companies looking to efficiently handle their data growth. Database performance in handling XML data is a key factor to consider when choosing a database platform on which to store your XML data.

This study demonstrated that IBM DB2 is clearly the leader in handling XML data. DB2 9.5 with pureXML showed unparalleled database performance in all scenarios.

In comparison to Microsoft SQL Server 2008, DB2 9.5 was significantly faster in all tests:

IBM DB2 9.5 was able to index and store XML data up to 150x faster than Microsoft SQL Server 2008 while using 1/10th the total disk space. IBM DB2 9.5 was able to query XML data from 2.5x up to 6x faster than Microsoft SQL Server 2008.

DB2 also outperformed Oracle 11g. IBM DB2 9.5 was able to index and store XML data from 6x up to 86x faster than Oracle 11g while using up to 1/3rd less disk space. IBM DB2 9.5 was able to query XML data from 2.5x up to 77x faster than with Oracle 11g.

IBM DB2's leading performance over Oracle 11g and Microsoft SQL Server 2008 gives you the best solution to store and access your XML and relational data and can save significant resources.

ACKNOWLEDGMENT

This paper is sponsored by IBM CDL China.

REFERENCES

- [1] Fatma Özcan, Normen Seemann, Ling Wang, XQuery Rewrite Optimization in IBM DB2 pureXML, IEEE Data Engineering Bulletin, 31(4):25-32, December 2008.
- [2] International Organization for Standardization (ISO). Information Technology-Database Language SQL-Part 14: XML-Related Specifications (SQL/XML), ANSI/ISO/IEC 9075-14:2006.
- [3] A. Balmin and F. Özcan and A. Singh and E. Ting. Grouping and optimization of XPath expressions in DB2 pureXML. In Proc. of SIGMOD, 2008.
- [4] XQuery 1.0: An XML Query Language, January 2007. W3C Recommendation, See <http://www.w3.org/TR/xquery>.
- [5] XQuery 1.0 and XPath 2.0 Data Model, January 2007. W3C Recommendation, See <http://www.w3.org/TR/xpath-datamodel>.
- [6] S. Pal et al. XQuery Implementation in a Relational Database System. In Proc. of VLDB, 2005.
- [7] K. Beyer et al. System RX: One part relational, one part XML. In Proc. of ACM SIGMOD, pages 347-358, 2005.
- [8] IBM red book: DB2 Version 9.1 for z/OS XML Guide (SC18-9858-05), 2007.3
- [9] IBM red book: DB2 Version 9.1 for z/OS XML Extender Administration and Programming (SC18-9857-00), 2007.3
- [10] IBM red book: DB2 Version 9.1 for z/OS SQL Reference (SC18-9854-06), 2009.5